



Blaise



Blaise 5

Fancy MVC Applications

A Life History Calendar

Bas Huijts
shj.huijts@cbs.nl

Agenda

- What is a Life History Calendar [LHC]
- Version developed with NatCen
- Customization
- Improved versions
- Q&A



What is a LHC

Year 71 72 73 74 75 76 77 78 79 80 81 82 83 84
Age 16 17 18 19 20 21 22 23 24 25 26 27 28 29

Education + Full time education including primary... +

Children - First child... + Second child (name)... +
Child 1 First child (name)... +
Child 2 Second child (name)... +
Child 3

Partners +

Accommodation + Living at (Address)... +

Personal events: Important events:

◀ 1971 - 1980 ▶

Year 71 72 73 74 75 76 77 78 79 80
Age 16 17 18 19 20 21 22 23 24 25

Education + Full time education including primary... +

Children - First child... + Second child (name)... +
Child 1 First child (name)... +
Child 2 Second child (name)... +
Child 3

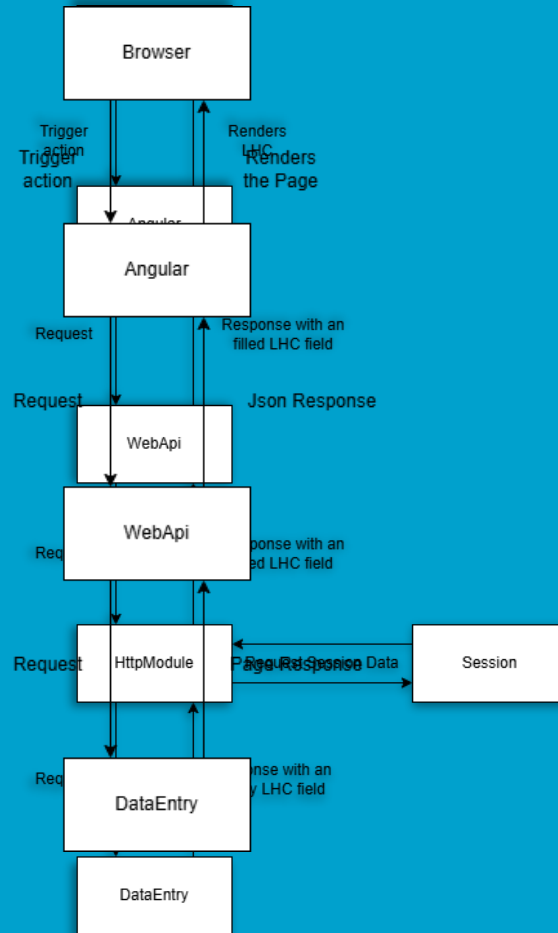
Partners +

Accommodation + Living at (Address)... +

- Visual representation of events in a respondents life
 - How many children
 - Which partners?
 - Where did the respondents live
- Based on blocks [topics] the data should be presented
- Should be updated during the survey
- Mobile version
- Important events as guideline for filling in the data
- Clicking on a year or event should display detailed information
- Selecting an event should go to the correct question



Version NatCen



- In the datamodel there is a field [LifeHistoryCalendar] to fill with all the needed data
- Use an HttpModule to retrieve the data from the session and fill the field in the response
- The Angular Runtime renders the calendar based on the data supplied in the field
- Using custom properties the field is put on the correct pages and can be customized as needed



Version NatCen

```
BLOCK B_CalendarInfo
  FIELDS
    name "Name of Event" : STRING
    description "Description of Event" : STRING
    startDate "Start Date of Event" : DATETIME
    endDate "End Date of Event" : DATETIME, EMPTY
  RULES
    name.keep
    description.keep
    startDate.keep
    endDate.keep
ENDBLOCK
```

```
BLOCK BChild
  PARAMETERS
    IMPORT piIndex: INTEGER
  FIELDS
    name "What is the name of the child?" : STRING[25]
    description "What is the description for this child?" : STRING[25]
    startDate "When was he/she born?" : DATETIME
    endDate "When died he/she die?" : DATETIME, EMPTY
    repeat "Add Another?" : TYesNo
  RULES
    name
    calendarInfo[piIndex].name := name
    calendarInfo[piIndex].name.MappedFieldName := 'Children[' + ToString(piIndex) + ']'
    description
    calendarInfo[piIndex].description := description
    calendarInfo[piIndex].description.MappedFieldName := 'Children[' + ToString(piIndex) + ']'
    startDate
```

```
Children : ARRAY [1..5] OF BChild
calendarInfo
  Topic "Children"
  TextRole_StartDate "Birth of child"
  TextRole_EndDate "Death of child"
  : ARRAY [1..5] OF B_CalendarInfo
```

```
// field to store the calendar data
LifeHistoryCalendar.KEEP

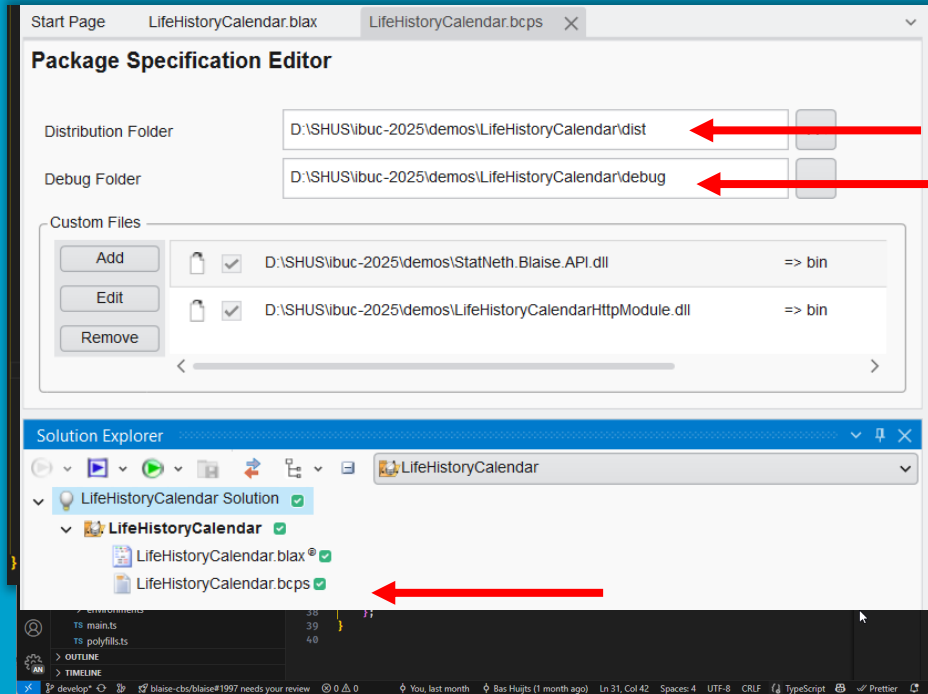
// start with child 1
// the loop should end with Topic
Children[1](1)
// loop while another child
FOR index := 2 TO 5 DO
  IF Children[index-1].repeat = yes THEN
    Children[index](index)
  ENDIF
ENDDO
```

Datamodel

- Used the sample 'LocalArrayModify' as starting point for this use case
- Basic block for all the topics/events
- Use this block for the specific topics and create mappedfields so we know in which topic we are working
- Create the necessary fields
- Ask the questions in an array till the respondent stops the loop
- Placed a control on the page for the field, bound the KEEP field to this control using a FIELD reference



Customize



- How do we customize Angular:
 - Use customproperties as hooks for Angular
 - Create the components we need
 - Make Angular use them
 - Tell Blaise to use the custom runtime



Customize

```
0 references | Bas Huijts, 3 days ago | 1 author, 4 changes
public class HttpModule : IHttpModule
{
    /// <summary>
```

```
0 references | Bas Huijts, 27 days ago | 1 author, 1 change
public void Init(HttpApplication context)
{
    Trace.TraceInformation("Elsa.HttpModule initialized");
    context.BeginRequest += Context_BeginRequest;
}
```

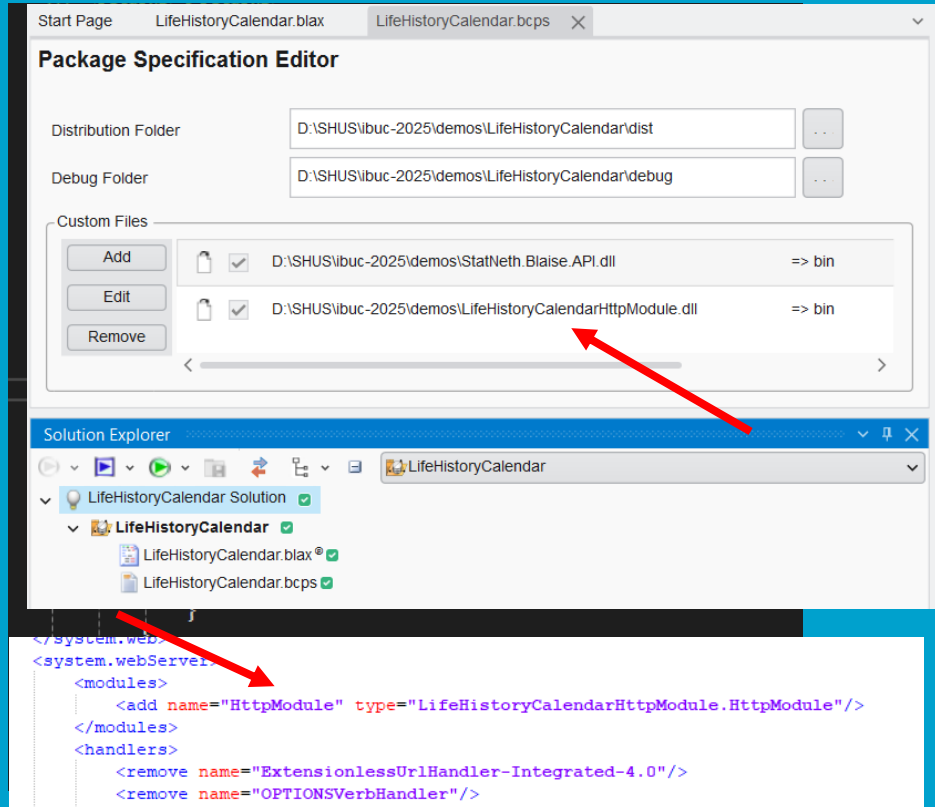
```
/// <param name="e"></param>
1 reference | Bas Huijts, 25 days ago | 1 author, 3 changes
private void Context_BeginRequest(object sender, EventArgs e)
{
    _app = sender as HttpApplication;
    _path = _app.Request.Path;

    // the old way
    if (_path.Contains("start_interview") || _path.Contains("ex
    {
        // create new generic response filter
        var start = DateTime.Now;
        Trace.TraceInformation("LifeHistoryCalendarHttpModule.H
        OutputFilterStream ofs = new OutputFilterStream(_app.Re
        var end = DateTime.Now;
        Trace.TraceInformation("LifeHistoryCalendarHttpModule.H
        // attach function to transform response string
        ofs.TransformString += TransformResponseString;
        // attach filter
        _app.Response.Filter = ofs;
    }
}
```

- How do we use a HttpModule:
 - Create an class which implements IHttpModule
 - Attach an eventhandler to BeginRequest
 - Use an OutputFilterStream to tap into the output stream
 - Create a function to transform the output string supplied to the browser



Customize



The screenshot shows the Visual Studio interface. The Package Specification Editor is open, showing the Distribution Folder and Debug Folder. The Custom Files section lists two files: D:\SHUS\ibuc-2025\demos\StatNeth.Blaise.API.dll and D:\SHUS\ibuc-2025\demos\LifeHistoryCalendar\HttpModule.dll. The Solution Explorer shows the LifeHistoryCalendar project. The web.config file is open, showing the system.webServer section with the HttpModule added.

```
</system.web>
<system.webServer>
  <modules>
    <add name="HttpModule" type="LifeHistoryCalendar.HttpModule.HttpModule"/>
  </modules>
  <handlers>
    <remove name="ExtensionlessUrlHandler-Integrated-4.0"/>
    <remove name="OPTIONSVerbHandler"/>
  </handlers>
</system.webServer>
```

- How do we use a HttpModule:
 - In the transform function we retrieve the session data, convert it to the correct format and put it back in the LHC field
 - Make sure Blaise deploys the dll file in IIS
 - Finally enable the HttpModule in the system.webserver section of the web.config



Demo

- Let's see the result of the version developed for NatCen



Conclusions

- After some development iterations we came to a production version of the LHC.
- Putting all the data in one field, has impact on the performance especially when using the old SizeTree render engine.
- As the survey gets larger, the amount of data in the LHC field grows, this blocks the page response since it is a synchronous action.



Improvement 1

- Asynchronous version
 - Retrieve the data async so the page doesn't get blocked
 - Some data will not change during the page, pr even during the survey so why not cache / retain it
 - Only the data for the current block can change

For demonstration purpose I only implemented it for the desktop



Improvement 1

```
// the async calls from the calendar only occur as an extension of the start_interview or executead
if (_path.Contains("start_interview") || _path.Contains("executeaction"))
{
    // general code for the calendar
    if (_path.Contains("lhc"))
    {
        var stream = _app.Request.InputStream;
        string requestBody;

        using (var reader = new System.IO.StreamReader(stream, Encoding.UTF8))
        {
            requestBody = reader.ReadToEnd();
            Object parsedJson = JObject.Parse(requestBody);

            var keyValue = parsedJson["keyvalue"].ToString();

            ISessionInfo sessionInfo = SessionInfoHelper.GetSessionInfo(keyValue);

            // initialize call of the calendar
            if (_path.Contains(Constants.ROUTE_INITIALIZE))
            {
                var mobile = parsedJson.SelectToken("mobile").Value<bool>();
                Decade currentDecade = parsedJson.SelectToken("decade") != null ? parse

                InitializeResult json = new InitializeResult(sessionInfo, mobile, curre
                _app.Response.Write(JsonConvert.SerializeObject(json, dateFormattingSe

            }
            else if (_path.Contains(Constants.ROUTE_EVENTS))
            {
                EventsResult json = new EventsResult(sessionInfo);
                _app.Response.Write(JsonConvert.SerializeObject(json, dateFormattingSe

            }
            else if (_path.Contains(Constants.ROUTE_EVENTS_FOR_TOPIC))
            {
                var topic = parsedJson["topic"];
                Decade currentDecade = parsedJson.SelectToken("decade") != null ? parse

                EventsForTopicResult json = new EventsForTopicResult(sessionInfo, int.I
                _app.Response.Write(JsonConvert.SerializeObject(json, dateFormattingSe

            }
            else if (_path.Contains(Constants.ROUTE_IMPORTANT_EVENTS))
            {
                ImportantEventsResult json = new ImportantEventsResult();
                _app.Response.Write(JsonConvert.SerializeObject(json, dateFormattingSe

            }
        }

        // end the response to prevent further bubbling of the event through the webserver
        _app.Response.End();
    }
}
```

- HttpModule
 - Extended the existing Blaise calls with an sub path “lhc”
 - Created separate endpoint for the data to be retrieved
 - Made sure when the sub calls are made the default call is ended



Improvement 1

```
public data$: Observable<LHC_Class_Async>;  
You, 23 seconds ago • Uncommitted changes
```

```
ngOnInit(mobile: boolean = false): void {  
  // call the base ngOnInit  
  super.ngOnInit();  
  
  this.data$ = this.service  
    .getInitializeData(  
      this.sp.renderStateService.layoutModelState.KeyValue.toString(),  
      mobile  
    );
```

```
[ngStyle]= style  
[ngClass]="cssClass">  
<div class="table overflow-auto" *ngIf="(data$ | async) as _data">  
  <table class="striped cell-border-bottom all-border-right">  
    <thead>  
      <tr>  
        <th><!-- Headers for decades, years, age -->
```

- Angular
 - Create an Observable property
 - Get the data async on initialize
 - Adjust the template to made them async aware

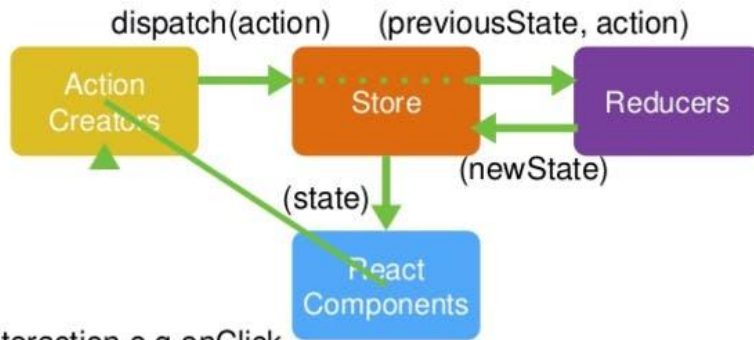


Demo



Improvement 2

Redux Flow



React + Redux

@nikgraf

- Use the Redux Pattern
 - Single source of truth: Store
 - Changes can only be done by dispatching an Action
 - Run the Reducer function to calculate the new state based on the old one.
 - Notify all subscribers to the state object



Improvement 2

```
@fortawesome/free-regular-svg-icons": "6.7.2",  
"@fortawesome/free-solid-svg-icons": "^6.7.2",  
"@ngrx/effects": "^19.0.1",  
"@ngrx/store": "^19.0.1",  
"@ngrx/store-devtools": "^19.0.1",  
"@syncfusion/ej2-angular-base": "^28.2.3",  
"@syncfusion/ej2-angular-richtext": "^28.2.3"
```

```
> services  
▼ state  
  TS lhc.actions.ts  
  TS lhc.effects.ts  
  TS lhc.reducers.ts  
  TS lhc.selectors.ts  
  app.component.scss  
  TS app.component.ts
```

```
//  
ages$ = this.store.select(selectAges);  
decades$ = this.store.select(selectDecades);  
loading$ = this.store.select(selectLoading);  
topics$ = this.store.select(selectTopics);  
dob$ = this.store.select(selectDOB);  
name$ = this.store.select(selectName);
```

- HttpClientModule – No changes needed
- Angular:
 - Add NgRx to project
 - Create the Actions, Effects, Reducers and Selectors
 - Use them in the components and templates



Demo



Recap

- Synchronous Version:
 - Sync calls block the page
 - Performance hit as the data grows
- Asynchronous Version:
 - Works better since all calls are async
 - Doesn't retain state
- NgRx/Redux Version:
 - Does retain state
 - Works async



Questions



A large, stylized blue logo is positioned on the left side of the slide. The logo consists of a white diamond shape with a blue border. Inside the diamond, there is a blue shape that resembles a stylized letter 'B' or a similar symbol. The background of the slide is split into a blue upper half and a white lower half.

**Thank you
for your time**

Have a nice IBUC 2025